**1. Tổng quan về Publisher**

* **Vai trò**: publisher.py chịu trách nhiệm lấy dữ liệu trạng thái của các Beacon từ Redis và gửi chúng lên AWS IoT Core thông qua giao thức MQTT. Đây là bước cuối cùng trong luồng dữ liệu để cập nhật trạng thái Beacon tới hệ thống đám mây.
* **Input**: Dữ liệu từ Redis queue aws\_queue (do processor.py tạo ra).
* **Output**: Dữ liệu trạng thái được gửi lên AWS IoT Core qua topic beacon/status.
* **Thư viện**: Sử dụng boto3 để giao tiếp với AWS IoT Core.

**2. Cách Publisher hoạt động**

**2.1. Khởi tạo và cấu hình**

* **Load biến môi trường**:
  + Publisher lấy thông tin cấu hình từ file .env bằng python-dotenv:

python

CollapseWrapCopy

load\_dotenv()

AWS\_REGION = os.getenv("AWS\_REGION", "us-west-2")

AWS\_IOT\_ENDPOINT = os.getenv("AWS\_IOT\_ENDPOINT", "your-iot-endpoint.amazonaws.com")

AWS\_TOPIC = os.getenv("AWS\_TOPIC", "beacon/status")

QUEUE\_HOST = os.getenv("QUEUE\_HOST", "localhost")

QUEUE\_PORT = int(os.getenv("QUEUE\_PORT", 6379))

* + Mặc định:
    - AWS Region: us-west-2.
    - AWS IoT Endpoint: Cần thay bằng endpoint thực tế của bạn (từ AWS IoT Core console).
    - Topic: beacon/status.
    - Redis: localhost:6379.
* **Kết nối Redis**:
  + Tạo kết nối tới Redis để lấy dữ liệu từ queue:

python

CollapseWrapCopy

redis\_client = redis.Redis(host=QUEUE\_HOST, port=QUEUE\_PORT, db=0)

* **Kết nối AWS IoT Core**:
  + Sử dụng boto3 để tạo client MQTT cho AWS IoT Core:

python

CollapseWrapCopy

aws\_client = boto3.client('iot-data', region\_name=AWS\_REGION, endpoint\_url=f"https://{AWS\_IOT\_ENDPOINT}")

* + iot-data: Dịch vụ AWS IoT Data Plane để publish dữ liệu qua MQTT.
  + Yêu cầu AWS credentials (Access Key, Secret Key) được cấu hình trong môi trường (qua AWS CLI hoặc file ~/.aws/credentials).

**2.2. Quy trình xử lý dữ liệu**

Publisher chạy trong một vòng lặp vô hạn để liên tục kiểm tra và gửi dữ liệu:

1. **Kiểm tra queue Redis**:
   * Dùng redis\_client.llen("aws\_queue") để kiểm tra xem queue aws\_queue có dữ liệu không:

python

CollapseWrapCopy

if redis\_client.llen("aws\_queue") > 0:

1. **Lấy dữ liệu từ queue**:
   * Lấy một phần tử từ đầu queue aws\_queue bằng lpop:

python

CollapseWrapCopy

data = redis\_client.lpop("aws\_queue")

if data:

data = json.loads(data.decode())

* + **Định dạng dữ liệu**: Chuỗi JSON từ Processor, ví dụ:

json

CollapseWrapCopy

{

"beacon\_id": "Beacon1",

"gateway\_id": "GW1",

"detected": 1,

"timestamp": "2025-03-11T10:00:00"

}

* + Nếu queue rỗng (data là None), bỏ qua và tiếp tục vòng lặp.

1. **Gửi dữ liệu lên AWS IoT Core**:
   * Publish dữ liệu lên topic beacon/status bằng boto3:

python

CollapseWrapCopy

aws\_client.publish(AWS\_TOPIC, json.dumps(data), qos=1)

* + AWS\_TOPIC: Lấy từ .env (mặc định beacon/status).
  + json.dumps(data): Chuyển dữ liệu JSON thành chuỗi.
  + qos=1: Quality of Service mức 1 (đảm bảo tin nhắn được gửi ít nhất một lần).

1. **In log**:
   * In thông báo lên console để debug:

python

CollapseWrapCopy

print(f"Sent to AWS: {data}")

1. **Delay vòng lặp**:
   * Ngủ 0.1 giây giữa các lần kiểm tra để tránh chiếm quá nhiều CPU:

python

CollapseWrapCopy

time.sleep(0.1)

**2.3. Chạy chương trình**

* **Hàm main**:
  + Chạy vòng lặp vô hạn trong main():

python

CollapseWrapCopy

def main():

while True:

if redis\_client.llen("aws\_queue") > 0:

data = redis\_client.lpop("aws\_queue")

if data:

data = json.loads(data.decode())

aws\_client.publish(AWS\_TOPIC, json.dumps(data), qos=1)

print(f"Sent to AWS: {data}")

time.sleep(0.1)

* **Kiểm tra chạy trực tiếp**:
  + Nếu file được chạy độc lập (\_\_name\_\_ == "\_\_main\_\_"), gọi main():

python

CollapseWrapCopy

if \_\_name\_\_ == "\_\_main\_\_":

main()

**3. Luồng hoạt động chi tiết**

1. **Khởi động**:
   * Publisher đọc .env, kết nối tới Redis và khởi tạo client AWS IoT Core.
2. **Kiểm tra queue aws\_queue**:
   * Liên tục kiểm tra xem có dữ liệu mới trong aws\_queue không.
3. **Lấy dữ liệu**:
   * Nếu có dữ liệu, lấy một phần tử từ đầu queue và parse JSON.
4. **Gửi lên AWS**:
   * Publish dữ liệu lên topic beacon/status qua AWS IoT Core.
   * In thông báo để xác nhận.
5. **Lặp lại**:
   * Ngủ 0.1 giây, sau đó quay lại bước 2.

**4. Ví dụ minh họa**

* **Dữ liệu trong aws\_queue**:

text

CollapseWrapCopy

{"beacon\_id": "Beacon1", "gateway\_id": "GW1", "detected": 1, "timestamp": "2025-03-11T10:00:00"}

* **Publisher xử lý**:
  + Lấy dữ liệu từ aws\_queue.
  + Gửi lên AWS IoT Core:
    - Topic: beacon/status
    - Payload: {"beacon\_id": "Beacon1", "gateway\_id": "GW1", "detected": 1, "timestamp": "2025-03-11T10:00:00"}
  + In console: Sent to AWS: {'beacon\_id': 'Beacon1', 'gateway\_id': 'GW1', 'detected': 1, 'timestamp': '2025-03-11T10:00:00'}

**5. Code đầy đủ của publisher.py**

python

CollapseWrapCopy

import json

import time

import os

import boto3

import redis

from dotenv import load\_dotenv

load\_dotenv()

AWS\_REGION = os.getenv("AWS\_REGION", "us-west-2")

AWS\_IOT\_ENDPOINT = os.getenv("AWS\_IOT\_ENDPOINT", "your-iot-endpoint.amazonaws.com")

AWS\_TOPIC = os.getenv("AWS\_TOPIC", "beacon/status")

QUEUE\_HOST = os.getenv("QUEUE\_HOST", "localhost")

QUEUE\_PORT = int(os.getenv("QUEUE\_PORT", 6379))

redis\_client = redis.Redis(host=QUEUE\_HOST, port=QUEUE\_PORT, db=0)

aws\_client = boto3.client('iot-data', region\_name=AWS\_REGION, endpoint\_url=f"https://{AWS\_IOT\_ENDPOINT}")

def main():

while True:

if redis\_client.llen("aws\_queue") > 0:

data = redis\_client.lpop("aws\_queue")

if data:

data = json.loads(data.decode())

aws\_client.publish(AWS\_TOPIC, json.dumps(data), qos=1)

print(f"Sent to AWS: {data}")

time.sleep(0.1)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**6. Tóm tắt cách hoạt động**

* **Kết nối**: Khởi tạo kết nối tới Redis và AWS IoT Core.
* **Lấy dữ liệu**: Liên tục kiểm tra và lấy dữ liệu từ aws\_queue trong Redis.
* **Gửi dữ liệu**: Publish dữ liệu trạng thái lên topic beacon/status qua AWS IoT Core.
* **Liên tục**: Chạy vòng lặp với delay 0.1 giây để xử lý dữ liệu mới.

**7. Yêu cầu để hoạt động**

* **AWS IoT Core**: Cần cấu hình endpoint và credentials trong AWS:
  + Tạo thing trong AWS IoT Core.
  + Lấy endpoint từ AWS Console (Settings > Device data endpoint).
  + Cấu hình AWS CLI hoặc file ~/.aws/credentials:

text

CollapseWrapCopy

[default]

aws\_access\_key\_id = YOUR\_ACCESS\_KEY

aws\_secret\_access\_key = YOUR\_SECRET\_KEY

* **Redis**: Phải chạy trên localhost:6379 (hoặc cấu hình khác trong .env).